Binus International

Database Technology:

Fluxium Hospital - Hospital Management System

Lecturer:

Dr. RAYMOND BAHANA, ST., M.Sc

Submitted by:
Christoffer Raffaelo Wijaya - 2602177051
Rafael Sutiono - 2602174535
Submitted to:

Dr. RAYMOND BAHANA, ST., M.Sc

Class: L3CC
Database Technology(COMP679901)

Faculty of Computer Science
Binus International University

Hospital Management System - Database Technology Project Proposal

What to submit:
A project proposal posts in your personal blog page, which contains:
1. **Project details:**
name of your project, your team, and team members.
2. **Problem statement:**
describe the problem that your proposed database system will solve and why do you need a database instead of any traditional file system? You can give a brief explanation about the business process of the user organization.
3. **Target user:**
HR(human resources)-department of any organization or business
who will use your database? who will administer the database? you are encouraged to give a real-life scenario.
4. **List of relations:**
identify at least 5 relations (tables) that you need to maintain for this database.

1. Project Details

- Project name :Fluxium Hospital - Hospital Management System
- Team name : CR - Fluxium
- Members :
    - Christoffer Raffaelo Wijaya (2602177051)
    - Rafael Sutiono (2602174535)

A Hospital Management System (HMS) is an all-inclusive software program created to streamline and mechanize various processes in a medical facility. This system integrates and manages different hospital functions in an effort to improve quality and efficiency of care. The fundamental functions of electronic health records, appointment scheduling, and patient registration facilitate department-to-department contact and data sharing. The system manages billing and invoicing, takes care of inpatient and outpatient care, and guarantees correct laboratory and pharmacy operations. Strict security protocols, financial reporting, and human resources management all help to create a well-organized hospital setting. Hospital administrators can make better decisions and achieve enhanced operational efficiency and patient care by implementing an HMS in addition to reducing paperwork. All things considered, a hospital management system is a crucial instrument in contemporary healthcare, encouraging resource management that is methodical and leading to a more efficient and patient-centered healthcare delivery system.

What Each Team Member Do:
Chris:
Database Administrator (DBA):

I make the DBA , which is responsible for setting up and maintaining the database server, ensuring proper configurations, and managing overall database health.
Setting SQL modes (SET SQL_MODE) and configuring server parameters.
Managing transactions (START TRANSACTION, SET AUTOCOMMIT).
Handling character set configurations. Not only that, but also I am involved in designing the database schema, defining tables, relationships, and constraints. Moreover, I Create tables (CREATE TABLE statements) such as admintb, appointmenttb, contact, doctb, patreg, and prestb. Specifying column data types, constraints, and primary keys.
Responsible for populating initial data into the tables, often done during system setup or migrations. Inserting sample data into tables (INSERT INTO statements) like admintb, appointmenttb, contact, doctb, patreg, and prestb. I developed the software application that interacts with the database, handling user inputs, and executing SQL queries.
Executing queries to retrieve, update, or insert data based on user actions.
Utilizing SQL statements within the application logic.

Raf:

Data Analyst or Report Developer:

I focused on extracting insights from the database and creating reports.
Responsibilities in the SQL script by formulating SQL queries to generate reports (e.g., patient appointments, contact messages) and extracting specific data for analysis.
I am also responsible for implementing and managing security measures to protect the database, which are Configuring user access (GRANT statements) and privileges, Setting up security-related parameters (SET statements), Quality Assurance (QA) Tester.
It verifies that the SQL statements function as expected, ensuring data integrity and system reliability. Testing SQL queries for correctness and efficiency. Ensuring that data is correctly inserted and retrieved. In charge of documenting the database structure, SQL queries, and usage guidelines. Documenting table structures, relationships, and constraints. Last but not least, providing comments or descriptions for complex queries.

2. Database Designs

A. Entity Relationships(ER)

Entities:

- Admintb
  Attributes: username (PK), password

- Appointmenttb
  Attributes: ID (PK), pid, fname, lname, gender, email, contact, doctor, docFees, appdate, apptime, userStatus, doctorStatus

- Contact
  Attributes: name, email, contact, message

- Doctb
  Attributes: username (PK), password, email, spec, docFees

- Patreg
  Attributes: pid (PK), fname, lname, gender, email, contact, password, cpassword

- Prestb
  Attributes: doctor, pid, ID, fname, lname, appdate, apptime, disease, allergy, prescription

Relationships:

- One-to-One:
  admintb to contact (via username)
  doctb to prestb (via username)
  patreg to prestb (via pid)
- One-to-Many:
  patreg to appointmenttb (via pid)
  doctb to appointmenttb (via username)
  patreg to prestb (via pid)

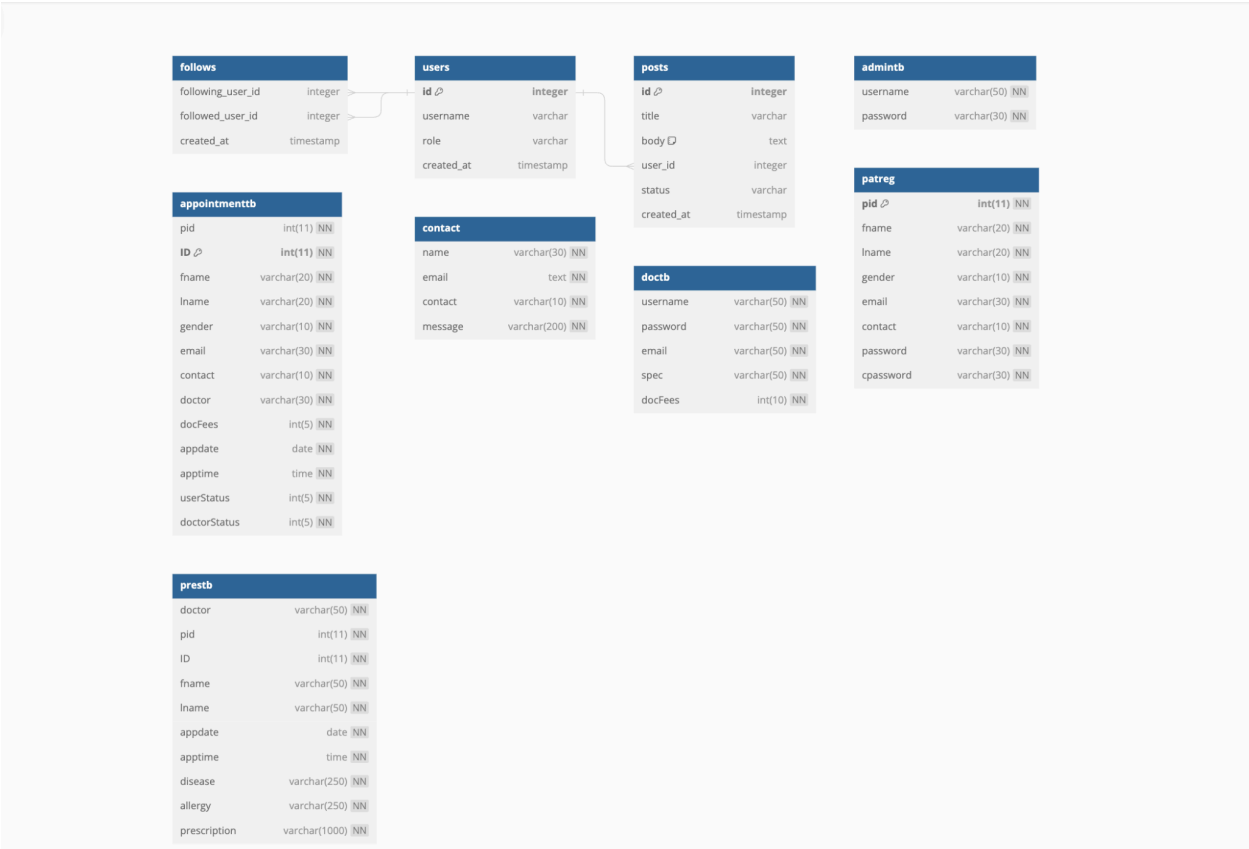
- Many-to-Many:
  There doesn't seem to be a direct many-to-many relationship between tables in the provided schema.

- Primary Keys:
  Primary keys are denoted by (PK).

- Foreign Keys:
  Foreign key relationships are not explicitly defined in the provided SQL dump. However, we can infer some relationships based on the column names.

- Possible foreign key relationships:
  appointmenttb.pid references patreg.pid
  appointmenttb.doctor references doctb.username
  prestb.pid references patreg.pid
  prestb.doctor references doctb.username

B. Relations



C. Normalisations

- ## 1NF(First Normal Form)
  Remove Duplicate Columns:

  No apparent duplicate columns in the provided schema.

- **Atomic Values:**

  Ensure that each column contains atomic (indivisible) values.

  Columns like ‘appointmenttb.data’ and ‘appointment tv.app time’ are atomic.

- ## 2NF(Second Normal Form)
  Remove Partial Dependencies:

- Break down tables that have composite primary keys.

- The ‘appointmenttb’ table has a composite primary key (ID and pid).
- Create a new table(appointments) with a surrogate primary key and move non-key attributes.

```
--
-- Table structure for table `appointmenttb`
--

CREATE TABLE `appointmenttb` (
  `pid` int(11) NOT NULL,
  `ID` int(11) NOT NULL,
  `fname` varchar(20) NOT NULL,
  `lname` varchar(20) NOT NULL,
  `gender` varchar(10) NOT NULL,
  `email` varchar(30) NOT NULL,
  `contact` varchar(10) NOT NULL,
  `doctor` varchar(30) NOT NULL,
  `docFees` int(5) NOT NULL,
  `appdate` date NOT NULL,
  `apptime` time NOT NULL,
  `userStatus` int(5) NOT NULL,
  `doctorStatus` int(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

- Remove Transitive Dependencies:
  Ensure that there are no transitive dependencies.

- The ‘appointmenttb’ table has a transitive dependency on ‘doctor’ for ‘docFees’.

- Create a new table (doctors) with ‘username’ as the primary key and move non-key attributes.

```
-
-- Dumping data for table `doctb`
-

NSERT INTO `doctb` (`username`, `password`, `email`, `spec`, `docFees`) VALUES
'ashok', 'ashok123', 'ashok@gmail.com', 'General', 500),
'arun', 'arun123', 'arun@gmail.com', 'Cardiologist', 600),
'Dinesh', 'dinesh123', 'dinesh@gmail.com', 'General', 700),
'Ganesh', 'ganesh123', 'ganesh@gmail.com', 'Pediatrician', 550),
'Kumar', 'kumar123', 'kumar@gmail.com', 'Pediatrician', 800),
'Amit', 'amit123', 'amit@gmail.com', 'Cardiologist', 1000),
'Abbis', 'abbis123', 'abbis@gmail.com', 'Neurologist', 1500),
'Tiwary', 'tiwary123', 'tiwary@gmail.com', 'Pediatrician', 450);

-- --------------------------------------------------------
```

Remove Non-Key Dependencies on Key:

- The ‘appoinmenttb’ table has non-key dependencies on ‘doctor’ for ‘docFees’.
- We have already addressed this by creating the ‘doctors’ table.

3. Sample Queries to Generate Report

It involves organizing and extracting the desired information using SQL queries. The sample queries are used below to create reports using the Hospital Management System database that is given. These inquiries address a number of topics, including prescription records, patient information, and appointment specifics.

1. List of All Appointments:

```
SELECT * FROM appointmenttb;
```

2. List of Patients With Contact Information:

```
SELECT pid, fname, lname, gender, email, contact FROM patreg;
```

3. Appointments for A Specific Doctor:

```
SELECT * FROM appointmenttb WHERE doctor = 'Christoffer';
```

4. **Total Doctor Fees Collected:**

```
SELECT SUM(docFees) AS TotalFees FROM appointmenttb;
```

5. **List of Contact Messages:**

```
SELECT * FROM contact;
```

6. **Prescriptions and Allergies for Patients:**

```
SELECT pid, fname, lname, disease, allergy, prescription FROM prestb;
```

7. **Appointments With User and Doctor Status:**

```
SELECT ID, fname, lname, appdate, apptime, userStatus, doctorStatus FROM appointmenttb;
```

8. **List of Doctor and Their Specifications:**

```
SELECT username, spec FROM doctb;
```

9. **Patients with Sever Fever:**

```
SELECT * FROM appointmenttb WHERE userStatus = 1 AND disease = 'Sever fever';
```

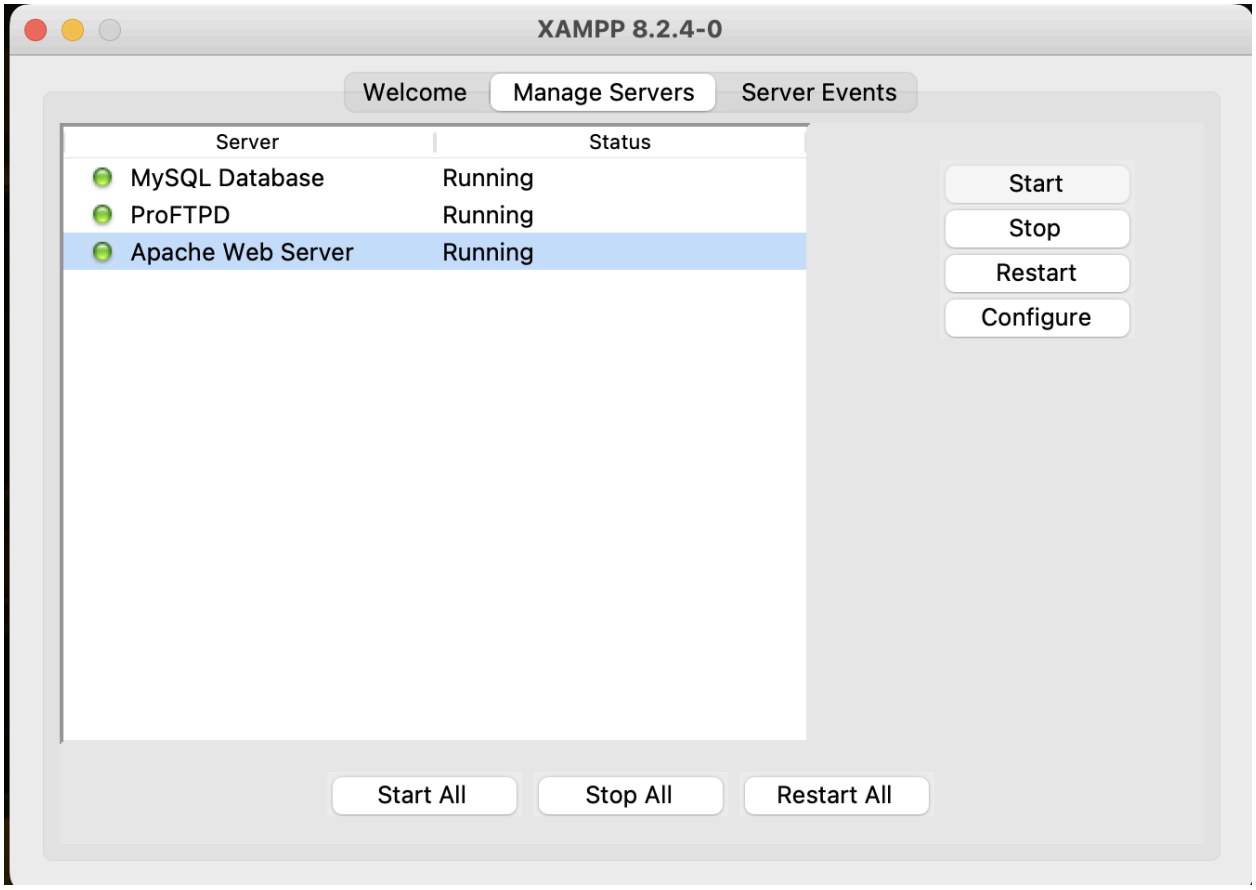
10. **Number of Appointments and Gender:**

```
SELECT gender, COUNT(*) AS AppointmentCount FROM appointmenttb GROUP BY gender;
```

4. User Interface

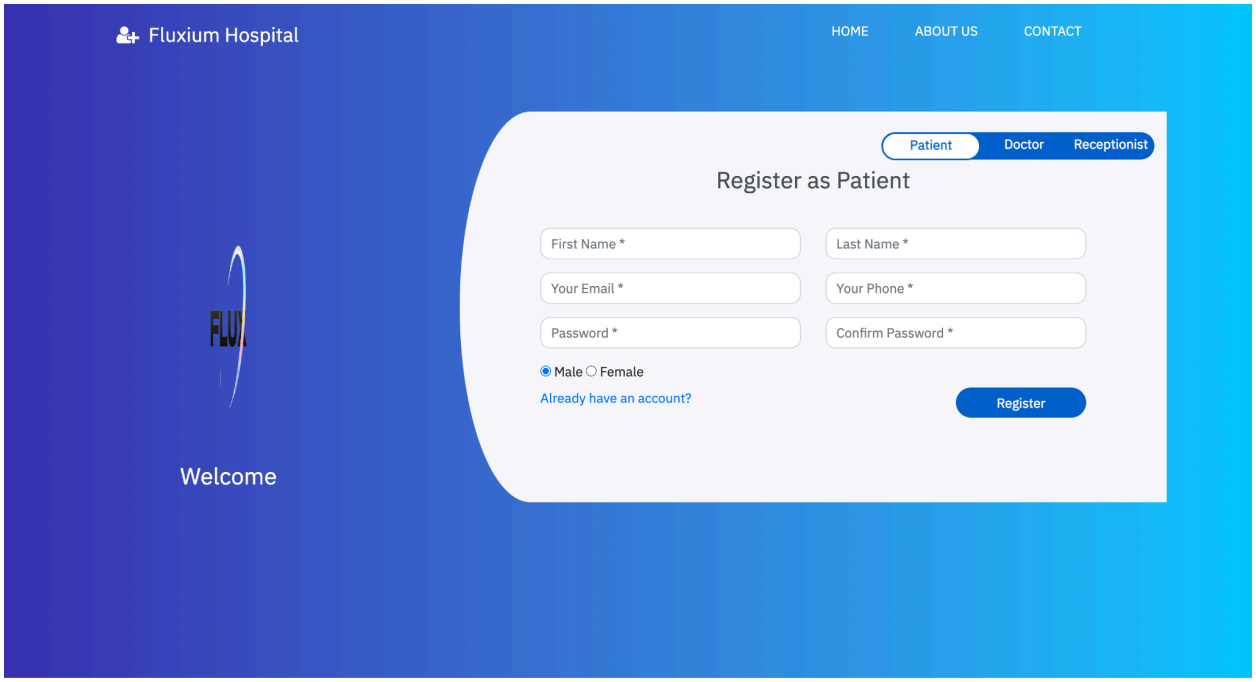
Starting Apache And MySQL in XAMPP:

The XAMPP Control Panel allows you to manually start and stop Apache and MySQL. To start Apache or MySQL manually, click the ‘Start’ button under ‘Actions’.

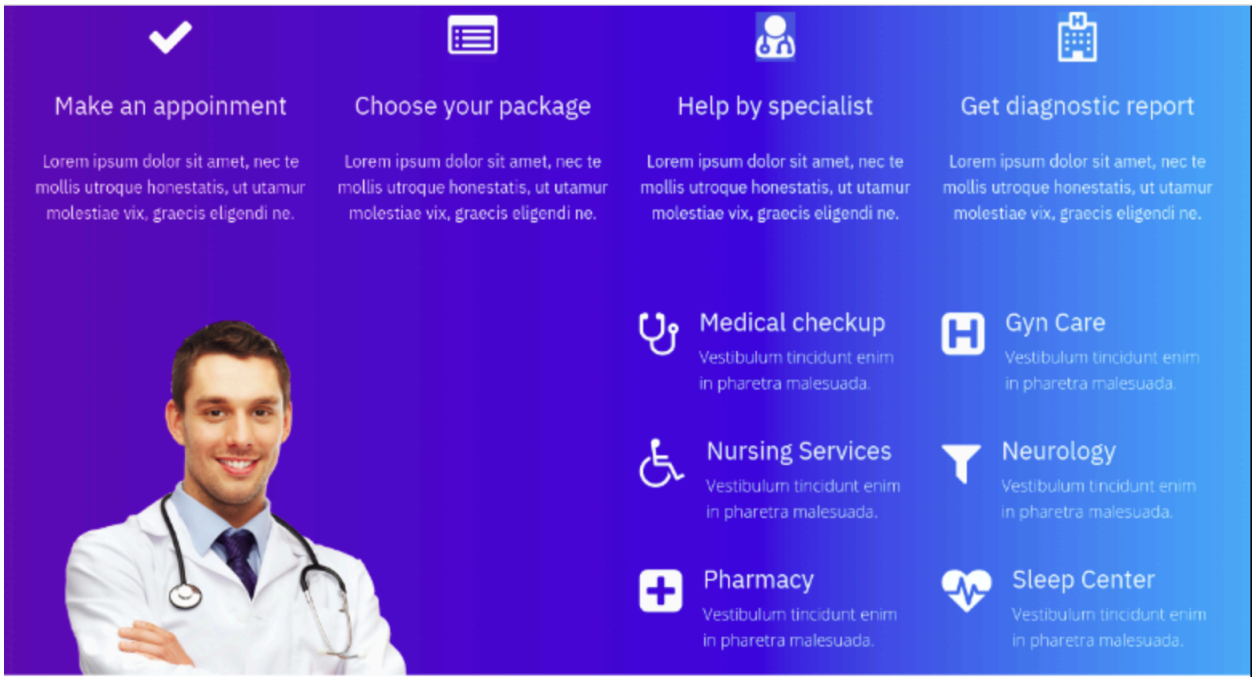


Getting into the project:

Hospital Management System in php and mysql. This system has a ‘Home’ page from where the patient, doctor & administrator can login into their accounts by toggling the tabs accordingly. The Picture below shows the ‘Home’ page of our project.



'About Us' page allows us to get some more information about the quality and the services of the hospital.



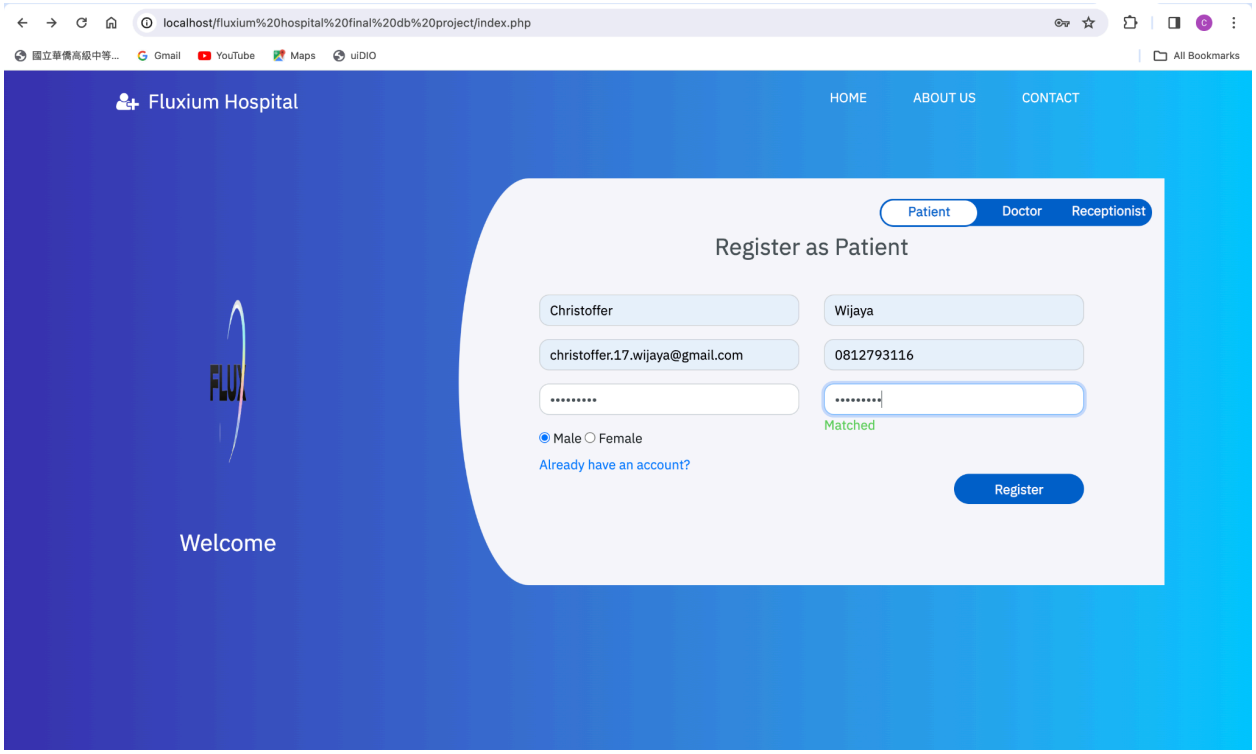
‘Contact’ page allows users to provide feedback or queries about the services of the hospital. Fig 1.3 shows the ‘Contact’ page.

The ‘Home’ page consists of 3 modules:

1. Patient Module
2. Doctor Module
3. Admin Module

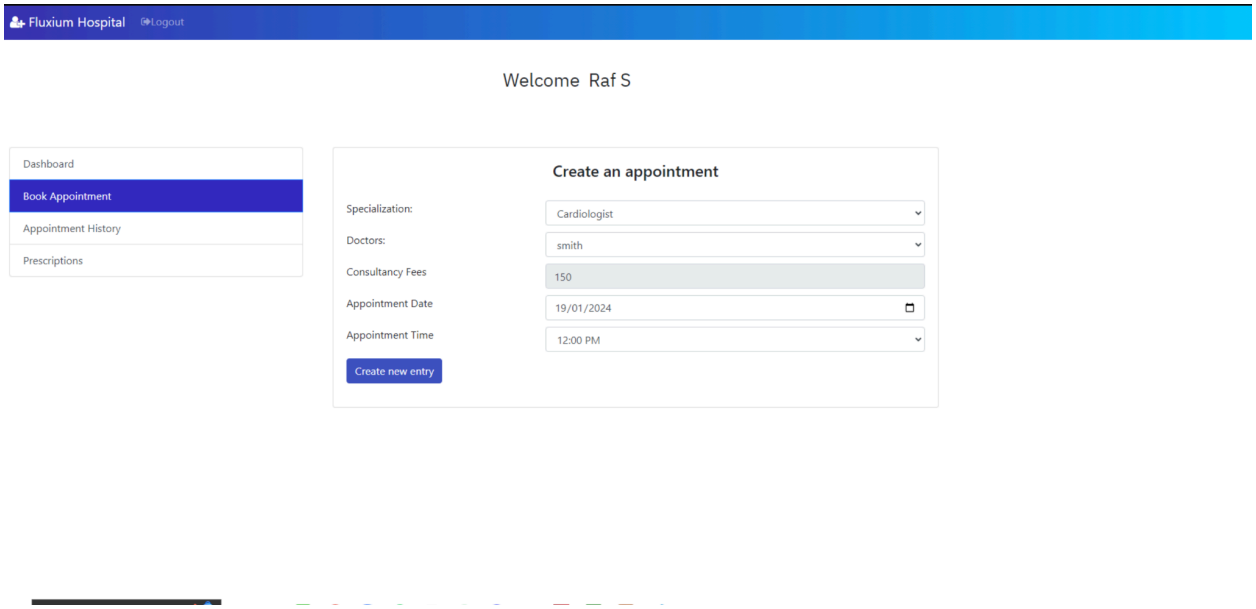
Patient Module:

This module allows patients to create their account, book an appointment to see a doctor and see their appointment history. The registration page(in the home page itself) asks patients to enter their First Name, Last Name, Email ID, Contact Number, Password and radio buttons to select their gender.

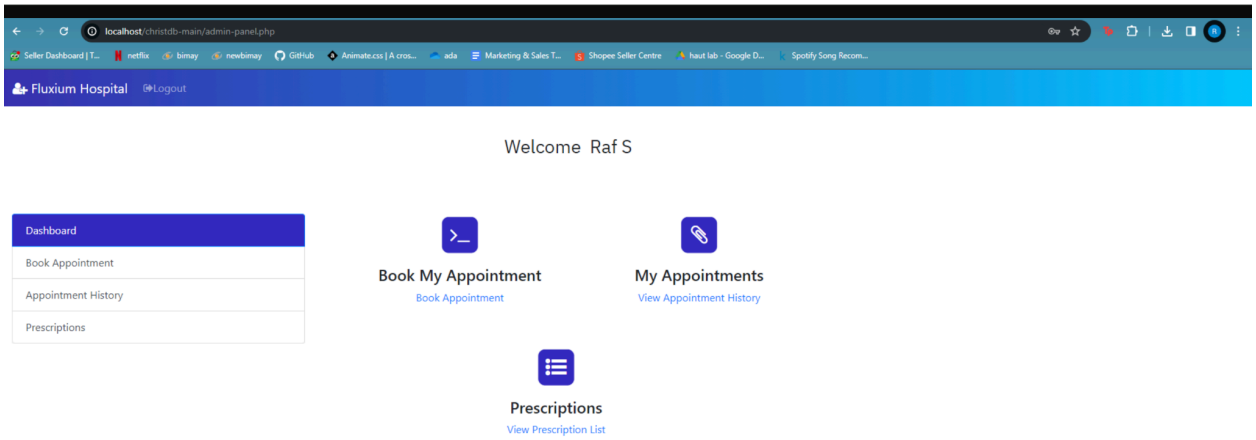


Book his/her Appointment:

Once the patient has created his/her own account after clicking the ‘Register’ button, then he will be redirected to his/her Dashboard. The image above is the dashboard page of our program. There are several dashboard functions such as Register section(FirstName, LastName, YourEmail, Your Phone, Password and ConfirmPassword), Appointment section(Dashboard, Book Appointment, Appointment History, Prescriptions),

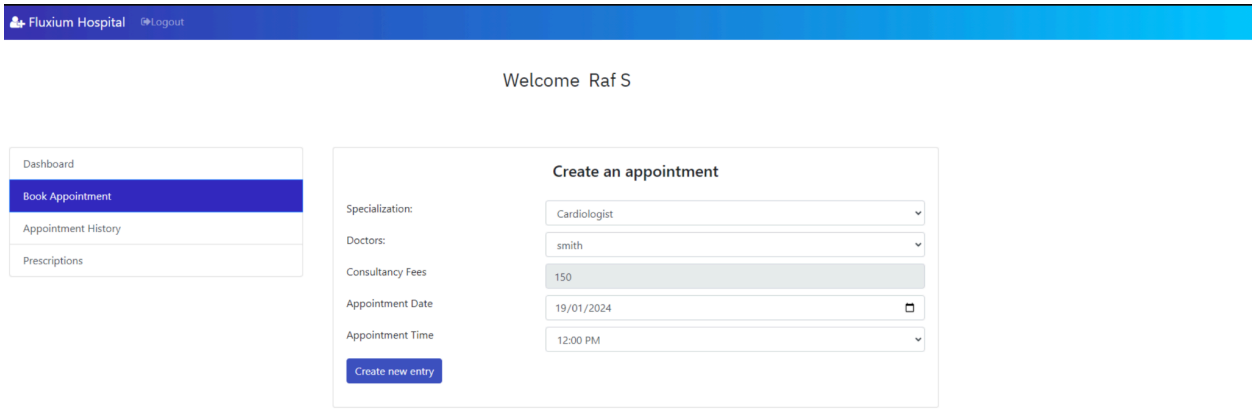


The Figure Below, After clicking on the 'Create new entry' button, the patient will receive an alert that acknowledges the successful appointment of the patient.



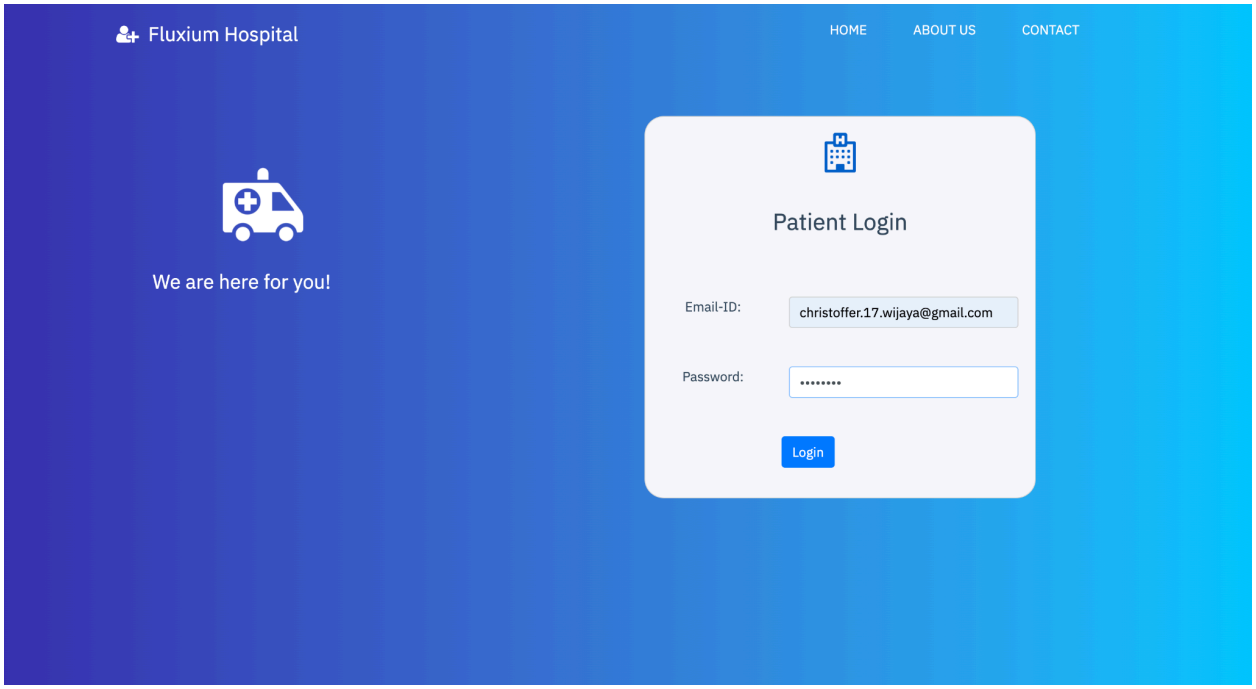
View Patients Appointment History:

Here, the patient can see their appointment history which contains Doctor Name, Consultancy Fee, Appointment Date and Time, the figure below.



View the list of patients appointments history:

Once the patient has logged out of his account, if he wants to go into his account again, he can login his account, instead of register his account again.The Picture below shows the login page. Clicking on 'Login' button will redirect the patient to his dashboard page which we have seen earlier.



This is how the patient module works. On the whole, this module allows patients to register their account or login their account(if he/she has one), book an appointment and view his/her appointment history.

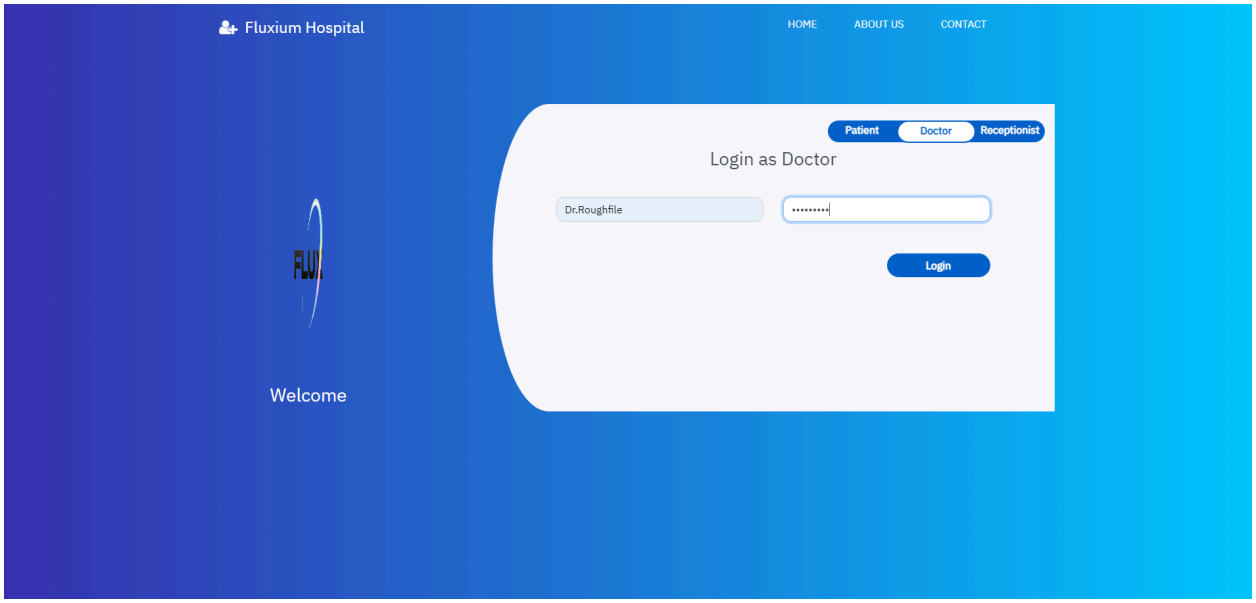
Doctor Module:

The doctors can login into their account which can be done by toggling the tab from ‘Patient’ to ‘Doctor’. The Figure below shows the login form for a doctor. Registration of a doctor account can be done only by admin. We will discuss more about this in Admin Module.

Once the doctor clicking the ‘Login’ button, they will be redirected to their own dashboard which is shown below.

In this page, doctor can able to see their appointments which has been booked by the patients. The Figure Below shows the appointment of the doctor ‘Christoffer’ which has been booked by the patient ‘Kenny Sebastian’ (Fig 1.6). This means that the doctor ‘Christoffer’ will have an appointment with the patient ‘Kenny Sebastian’ on 10-10-2023 10:15AM.

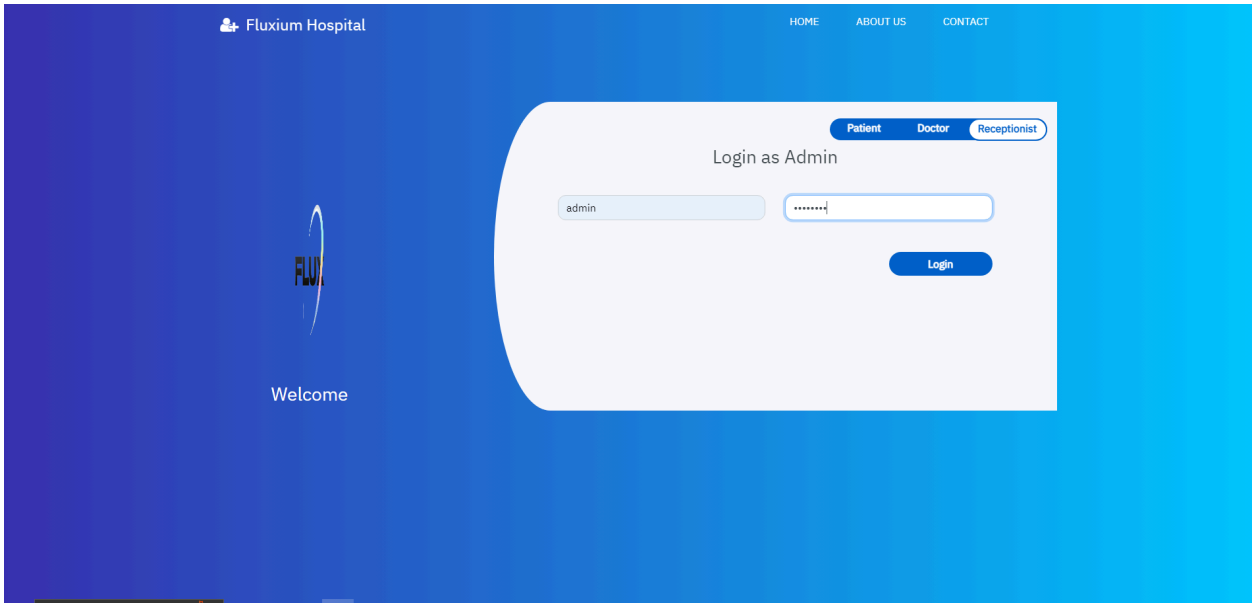
In real-time, the doctors will have thousands of appointments. It will be easier for a doctor to search for appointment in the case of more appointments. To make it easier, I have a ‘Search’ box in the navigation bar , in the figure below which allows doctors to search for a patient by their contact number. Once everything is done, the doctor can logout of their account. Thus, in general, a doctor can login into his/her account, view their appointments and search for a patient. This is all about Doctor Module.



Admin Module:

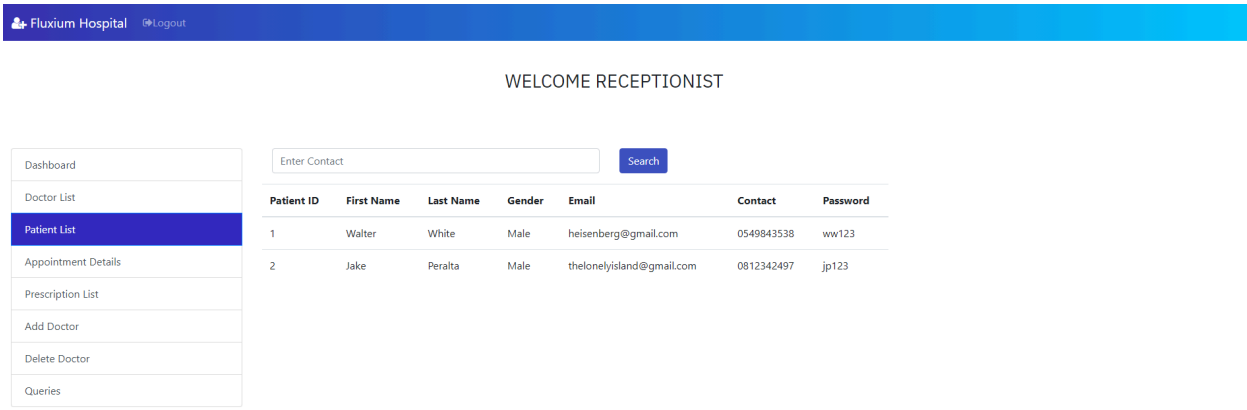
This module is the heart of our project where an admin can see the list of all patients. Doctors and appointments and the feedback/queries received from the 'Contact' page. Also admin can add doctor too. Login into admin account can be done by toggling into admin tab of the Home page. The Figure Below shows the login page for admin. Username: admin, Password: admin123

On clicking the 'Login' button, the admin will be redirected to his/her dashboard as shown in the figure below.



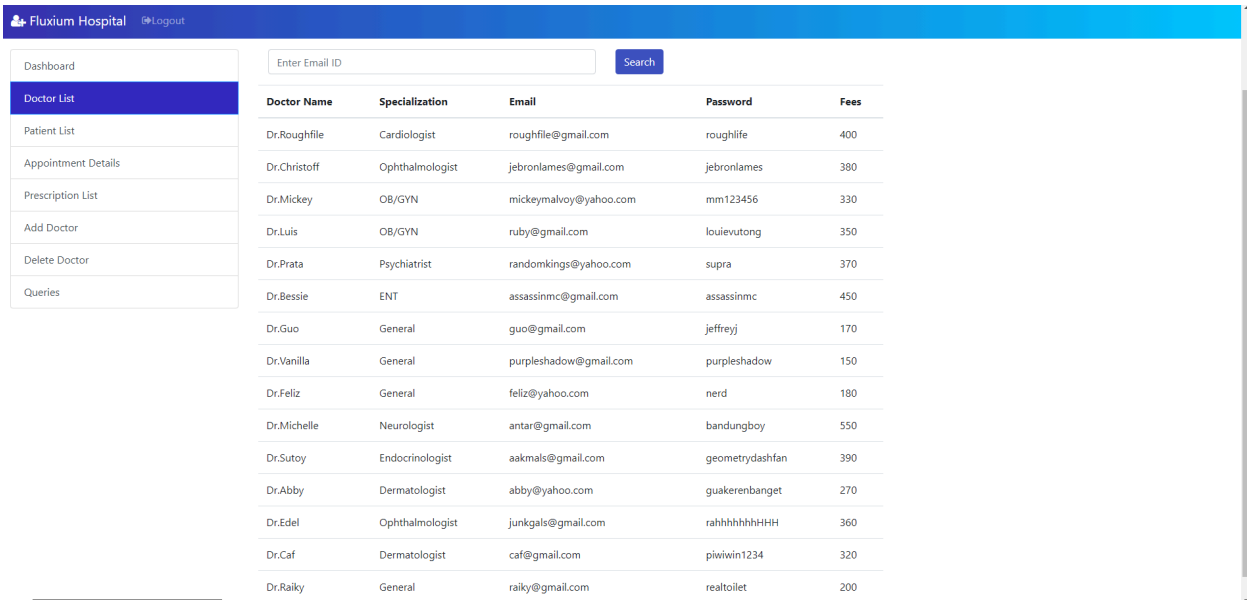
This module allows admin to perform five major operations:

- **View the list of all patients registered:**
  Admin can able to view all the patients registered. This includes the patients' First Name, Last Name, Email ID, Contact Number and Password. (See the figure below). As like in doctor module, admin can also search for a patient by their contact number in the search box.



- **View the list of all doctors registered:**

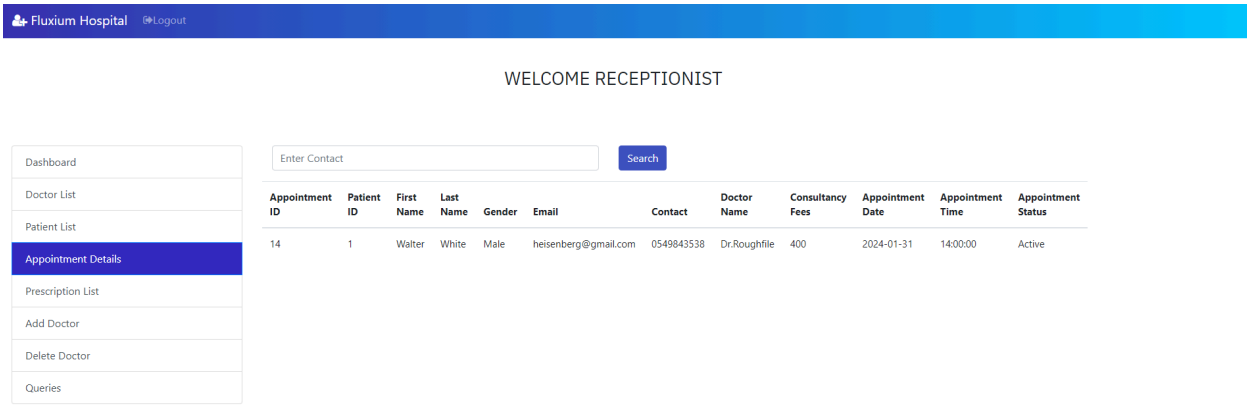
  Details of the doctors can also be viewed by the admin. This details include the Name of the doctor, Password, Email and Consultancy fees, shown in Fig 1.16. Searching for a doctor can be done by using the doctor's Email ID in the search box.



- **View the Appointment List:**

  Admin can also able to see the entire details of the appointment that shows the appointment details of the patients with their respective doctors. This includes the

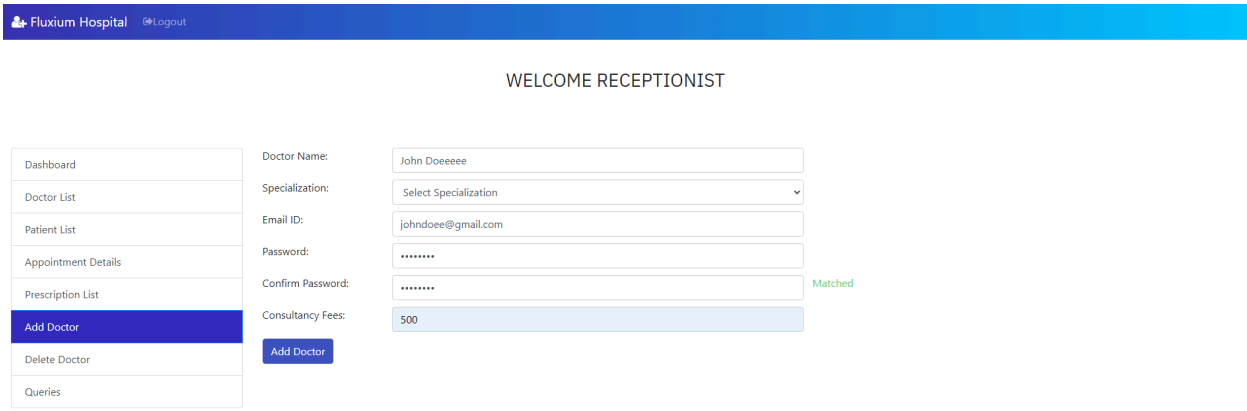
First Name, Last Name, Email and Contact Number of patients, doctor's name, Appointment Date, Time and the Consultancy Fees. (See the figure below).



- **Add Doctor**

  Admin alone can add a new doctor since anyone can register as a doctor if we put this section on the home page. This form asks Doctor's Name, Email ID, Password and his/her Consultancy Fees.(See the figure below).

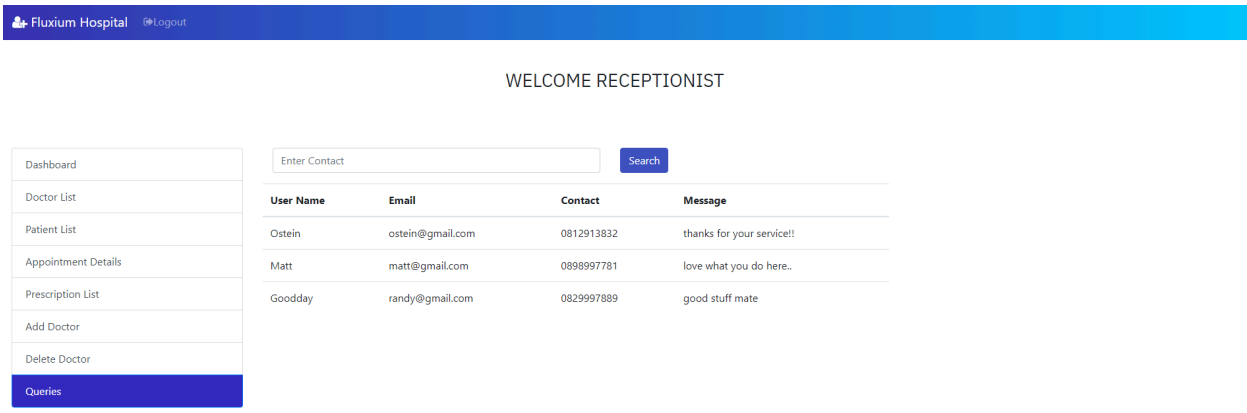
  After adding a new doctor, if we check the doctor's list, we will see the details of new doctor is added to the list as shown in the Figure below.



- **View User's Feedback**

  Admin is allowed to view the feedback/Query that has been given by the user in the 'Contact' page. This includes User's Name, Email Id, Contact Number and the message(Feedback/ Query) as shown in the Figure below

  Taking everything into consideration, admin can able to view the details of patients and doctors, appointment details, Feedback by the user and can add a new doctor. Once everything is done, the admin can logout from his account.



5. Database Security

Database security is a critical aspect of any information system, and the provided SQL script for the Hospital Management System (HMS) has a few security considerations. Here are some recommendations for enhancing database security:

User Authentication and Authorization:

Ensure that strong authentication mechanisms are in place for database users.

Implement the principle of least privilege, granting users only the necessary permissions required for their tasks.
Regularly review and update user roles and permissions.

Secure Password Storage:
Enforce the use of strong passwords for user accounts.
Hash and salt passwords before storing them in the database to enhance password security.

SSL/TLS Encryption:
Use SSL/TLS encryption to secure data in transit between the application and the database server.
Configure the database server to only accept encrypted connections.

SQL Injection Prevention:
Sanitize and validate user inputs to prevent SQL injection attacks.
Use parameterized queries or prepared statements in application code to mitigate the risk of SQL injection.

Regular Backups:
Implement regular database backups and store them securely.
Test the restoration process to ensure the availability of backups in case of data loss or corruption.

Audit Trails:
Enable database logging and auditing features to track user activities and changes to sensitive data.
Regularly review audit logs for suspicious activities.
Update and Patching:
Keep the database management system (DBMS) software up to date with the latest security patches.
Regularly check for updates and apply patches to address known vulnerabilities.

Firewall Rules:
Configure firewall rules to restrict access to the database server, allowing only necessary connections.
Whitelist specific IP addresses or network ranges that are allowed to connect to the database.

Secure Configuration:

Follow best practices for securing the database server's configuration.
Disable unnecessary database services and features that are not required for the application.

Error Handling:
Implement proper error handling in the application code to avoid exposing sensitive information in error messages.
Log errors securely without revealing critical details to users.

Data Encryption:
Consider encrypting sensitive data at rest using database-level encryption mechanisms.
Implement transparent data encryption (TDE) if supported by the DBMS.

Multi-Factor Authentication (MFA):
Implement multi-factor authentication for administrative accounts to add an extra layer of security.

Link to Github:
https://github.com/rafaelsutiono/hospital-management-system.git

Link to Google Docs:
https://docs.google.com/document/d/1puNp16A6mSNLs8yn4qvpVB1fjS2OTHIxpbHvv2RljwU/edit?usp=sharing

Link to Member's Personal Blog(Rafael Sutiono):

https://rafaelsutiono5.wordpress.com